Supplement IV.H: SpringLayout

For Introduction to Java Programming

Y. Daniel Liang

SpringLayout is a new Swing layout manager introduced in JDK 1.4. The idea of SpringLayout is to put a flexible spring around a component. The spring may compress or expand to place the components in desired locations.

To create a SpringLayout, use its no-arg constructor:

```
public SpringLayout()
```

A spring is an instance of the Spring class that can be created using one of the following two static methods:

- `public static` Spring constant(`int` pref)
  Returns a spring whose minimum, preferred, and maximum values each have the value pref.

- `public static` Spring constant(`int` min, `int` pref, `int` max)
  Returns a spring with the specified minimum, preferred, and maximum values.

Each spring has a preferred value, minimum value, maximum value, and actual value. The getPreferredValue(), getMinimumValue(), getMaximumValue(), and getValue() methods retrieve these values. The setValue(int value) method can be used to set an actual value.

The Spring class defines the static sum(Spring s1, Spring s2) to produce a combined new spring, the static minus(Spring s) to produce a new spring running in the opposite direction, and the static max(Spring s1, Spring s2) to produce a new spring with larger values from s1 and s2.

To add a spring or a fixed space to separate components in the container, use one of the following two methods in the SpringLayout class:

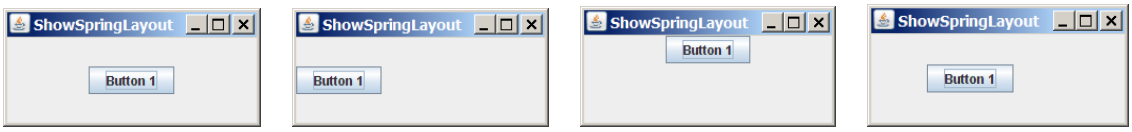
- `public void` putConstraint(String e1, Component c1, Spring s, String e2, Component c2)
  Places a spring between edge e1 of component c1 and edge e2 of component c2, anchored on c2. Each edge must have one of the following values: SpringLayout.NORTH, SpringLayout.SOUTH, SpringLayout.EAST, SpringLayout.WEST.

- `public void` putConstraint(String e1, Component c1, int pad, String e2, Component c2)
  Places a fixed pad between edge e1 of component c1 and edge e2 of component c2, anchored on c2.

In the preceding methods, e2 and c2 are referred to as the

anchor edge and *anchor component*, and e1 and c1 as the *dependent edge* and *dependent component*.

Listing 31.6 gives an example that places a button in the center of the container, as shown in Figure 31.11.



(a) (b) (c) (d)

Figure 31.11

The button is placed in the container of SpringLayout.

Listing 31.6 ShowSpringLayout.java

******PD: Please add line numbers in the following code*****
******Layout: Please layout exactly. Don't skip the space. This is true for all source code in the book. Thanks, AU.***
<Side Remark line 6: spring layout>
<Side Remark line 11: spring>
<Side Remark line 12: spring constraints>
<Side Remark line 23: main omitted>

```
import java.awt.*;
import javax.swing.*;

public class ShowSpringLayout extends JApplet {
  public ShowSpringLayout() {
    SpringLayout springLayout = new SpringLayout();
    JPanel p1 = new JPanel(springLayout);
    JButton jbt1 = new JButton("Button 1");
    p1.add(jbt1);

    Spring spring = Spring.constant(0, 1000, 2000);
    springLayout.putConstraint(SpringLayout.WEST, jbt1, spring,
                               SpringLayout.WEST, p1);
    springLayout.putConstraint(SpringLayout.EAST, p1, spring,
                               SpringLayout.EAST, jbt1);
    springLayout.putConstraint(SpringLayout.NORTH, jbt1, spring,
                               SpringLayout.NORTH, p1);
    springLayout.putConstraint(SpringLayout.SOUTH, p1, spring,
                               SpringLayout.SOUTH, jbt1);

    add(p1, BorderLayout.CENTER);
  }
        }
```

A SpringLayout named springLayout is created in line 6 and is set in JPanel p1 (line 7). An instance of Spring is created using the static constant method with minimum value 0, preferred value 1000, and maximum value 2000 (line 11).

Like icons and borders, an instance of Spring can be shared.

The putConstraint method in SpringLayout puts a spring between two components. In lines 12-13, the spring is padded between the west of the button and the west of the panel p1, anchored at p1. In lines 14-15, the same spring is padded between the east of the panel and the east of the button, anchored at the button. The selection of the anchor components is important. For example, if lines 14-15 were replaced by the following code,

```
    springLayout.putConstraint(SpringLayout.WEST, p1, spring,
                               SpringLayout.WEST, jbt1);
```

the button would be pushed all the way to the east, as shown in Figure 31.11(b).

If lines 18-19 were replaced by the following code,

```
    springLayout.putConstraint(SpringLayout.NORTH, p1, spring,
                               SpringLayout.NORTH, jbt1);
```

the button would be pushed all the way to the top, as shown in Figure 31.11(c).

If lines 14-15 were replaced by the following code,

```
    springLayout.putConstraint(SpringLayout.EAST, p1,
      Spring.sum(spring, spring), SpringLayout.EAST, jbt1);
```

the spring on the right of the button is twice strong as the left spring of the button, as shown in Figure 31.11(d).